|  |  |
| --- | --- |
| logomobile.png | BỘ GIÁO DỤC VÀ ĐÀO TẠO  **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM** |

**ĐỒ ÁN JAVA**

**Ứng Dụng Mua Hàng Trực Tuyến Trên Android**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Giảng viên hướng dẫn : Thanh Văn

Sinh viên thực hiện :

Nguyễn Mai Trường Phát

MSSV: 1811060560 Lớp: 18DTHB3

Phạm Phi Phong

MSSV: 1811062517 Lớp: 18DTHB3

Phạm Phú Lợi

MSSV: 1811061156 Lớp: 18DTHB3

TP. Hồ Chí Minh, 2021

LỜI CAM ĐOAN

Tôi xin cam đoan đồ án java “Ứng dụng mua hàng trực tuyến trên android” là công trình nghiên cứu của nhóm. Những phần sử dụng tài liệu tham khảo trong đồ án đã được nêu rõ trong phần tài liệu tham khảo. Các số liệu, kết quả trình bày trong đồ án là hoàn toàn trung thực, nếu sai tôi xin chịu hoàn toàn trách nhiệm và chịu mọi kỷ luật của bộ môn và nhà trường đề ra.

Sinh viên thực hiện

**Nguyễn Mai Trường Phát**

**Phạm Phi Phong**

**Phạm Phú Lợi**

# LỜI MỞ ĐẦU

Ngày nay Internet đã trở thành dịch vụ phổ biến và thiết yếu và có ảnh hưởng sâu rộng tới thói quen, sinh hoạt, giải trí của nhiều người. Cùng với sự phát triển nhanh chóng của Internet thì các hình thức mua và bán hàng hóa cho mọi người ngày càng đa dạng và phát triển hơn. Các ứng dụng điện thoại ngày càng trở nên phổ biến. Trước nhu cầu đó, cùng với yêu cầu môn học, nhóm chúng em quyết định chọn đề tài  **Ứng dụng mua hàng trực tuyến trên android**.

Với đề tài và môn học này, em xin chân thành cảm ơn sự giúp đỡ tận tình của thầy **Thanh Văn**. Song, do còn nhiều hạn chế đề tài nhóm xây dựng không tránh khỏi những thiếu sót. Rất mong được thầy cô và các bạn đóng góp ý kiến để chương trình ngày càng hoàn thiện và được đưa vào sử dụng.?

**Trân trọng cảm ơn!**

Sinh viên thực hiện

**Nguyễn Mai Trường Phát**

**Phạm Phi Phong**

**Phạm Phú Lợi**

MỤC LỤC

Mục Lục

[LỜI MỞ ĐẦU 3](#_Toc68712214)

[CHƯƠNG 1. Tổng Quan Hệ Thống 6](#_Toc68712215)

[1.1. Mô tả yêu cầu khách hàng. 6](#_Toc68712216)

[1.2 . Phân tích yêu cầu chức năng. 6](#_Toc68712217)

[1.2.1 Phân tích yêu cầu chức năng. 6](#_Toc68712218)

[1.2.2 Phân tích yêu cầu phi chức năng. 7](#_Toc68712219)

[1.2.3 Quy tắc nghiệp vụ. 7](#_Toc68712220)

[1.2.4 Quy trình nghiệp vụ. 7](#_Toc68712221)

[1.3 . Mô hình hóa yêu cầu chức năng. 8](#_Toc68712222)

[1.3.1 Sơ đồ use case. 8](#_Toc68712223)

[1.3.2 Đặc tả use case. 8](#_Toc68712224)

[1.3.3 Sơ đồ sequence. 15](#_Toc68712225)

[1.3.4 Sơ đồ class. 20](#_Toc68712226)

[CHƯƠNG 2. Cơ Sở Dữ Liệu 20](#_Toc68712227)

[2.1. Giới thiệu tổng quan về Firebase. 20](#_Toc68712228)

[2.1.1. Firebase là gì? 20](#_Toc68712229)

[2.1.2. Lịch sử phát triển của FireBase. 21](#_Toc68712230)

[2.1.3. Những dịch vụ nổi bật 22](#_Toc68712231)

[2.1.3.1. Realtime Database 23](#_Toc68712232)

[2.1.3.2. Authenticate 24](#_Toc68712233)

[2.1.3.3. Firebase Cloud Messaging(FCM). 25](#_Toc68712234)

[2.1.3.4. Firebase Database Query 26](#_Toc68712235)

[2.1.3.5. Remote Config 27](#_Toc68712236)

[2.2. Cơ sở dữ liệu của ứng dụng. 28](#_Toc68712237)

[2.2.1. Tổng quan cơ sở dữ liệu của ứng dụng. 28](#_Toc68712238)

[2.2.2. Một số thành phần dữ liệu 29](#_Toc68712239)

[2.2.2.1. Dữ liệu danh mục. 29](#_Toc68712240)

[2.2.2.2. Dữ liệu thương hiệu. 29](#_Toc68712241)

[2.2.2.3. Dữ liệu tài khoản. 29](#_Toc68712242)

[2.2.2.4. Dữ liệu sản phẩm. 30](#_Toc68712243)

[2.3. Mã hóa dữ liệu 31](#_Toc68712244)

[2.3.1. Tại sao không nên lưu plain-text, encrypt hoặc dùng MD5, SHA-1 31](#_Toc68712245)

[2.3.2. Tại sao phải Salt 31](#_Toc68712246)

[2.3.3. Hash ở đâu. 32](#_Toc68712247)

[2.3.4. Tại sao dùng bcrypt thay cho SHA-512 33](#_Toc68712248)

[2.3.5. Tại sao cần pepper 33](#_Toc68712249)

[CHƯƠNG 3. Kết Quả Thực Nghiệm. 35](#_Toc68712250)

[3.1. Thiết kế giao diện hệ thống. 35](#_Toc68712251)

[3.1.1. Giao diện trang chủ 35](#_Toc68712252)

[3.1.2. Giao diện yêu thích 36](#_Toc68712253)

[3.1.3. Giao diện giỏ hàng 37](#_Toc68712254)

[3.1.4. Giao diện xác nhận đơn hàng 38](#_Toc68712255)

[3.1.5. Giao diện sản phẩm 39](#_Toc68712256)

[3.1.6. Giao diện tìm kiếm 40](#_Toc68712257)

[3.1.7. Giao diện đăng nhập 41](#_Toc68712258)

[3.1.8. Giao diện đăng ký 42](#_Toc68712259)

[3.1.9. Giao diện tài khoản cá nhân 43](#_Toc68712260)

[3.2. Testcase của ứng dụng. 44](#_Toc68712261)

# CHƯƠNG 1. Tổng Quan Hệ Thống

## 1.1. Mô tả yêu cầu khách hàng.

Xây dựng một hệ thống bán hàng trực tuyến trên nền tảng ứng dụng di động android đơn giản, thân thiện, dễ sử dụng, cho phép khách hàng xem thông tin và đặt hàng qua ứng dụng, người quản trị quản lý các thông tin về sản phẩm. Đối với khách hàng, có thể xem và tìm kiếm thông tin về sản phẩm có trên hệ thống, thêm xóa giỏ hàng,thêm xóa yêu thích. Sau khi đăng ký thành thành viên, khách hàng sẽ được đăng nhập vào ứng dụng để thực hiện đặt hàng thông qua giỏ hàng của mình. Khách hàng sẽ thanh toán trực tiếp sau khi nhận hàng, trước đó qua ứng dụng khách hàng đã thực hiện nhập đầy đủ thông tin giao hàng : họ tên, số điện thoại, địa chỉ giao hàng. Đối với người quản lý sẽ là người tiếp nhận đơn hàng, thực hiện việc giao hàng và xác nhận việc giao hàng thông qua người giao hàng. Đồng thời người quản lý sẽ là người kiểm soát cập nhật sản phẩm, xem thống kê, quản lý đơn đặt hàng.

## . Phân tích yêu cầu chức năng.

### Phân tích yêu cầu chức năng.

#### 1.2.1.1. Nghiệp vụ.

* Lưu trữ : ghi nhận thông tin đơn đặt hàng, thông tin người dùng, thông tin sản phẩm,...
* Tra cứu : Tìm và xem chi tiết sản phẩm có trên hệ thống
* Kết xuất : Lập báo cáo thông kê về doanh thu theo loại trong năm.

#### 1.2.1.2. Hệ thống.

* Môi Trường : số lượng sản phẩm
* Tự động : tiến trình giao hàng

### Phân tích yêu cầu phi chức năng.

#### 1.2.2.1. Liên quan đến người dùng.

* Tính tiện dụng : thông tin đăng ký dạng form, báo lỗi khi nhập sai , theo dõi được tiến trình giao hàng,...
* Tính hiệu quả

#### 1.2.2.2. Liên quan đến chuyên viên tin học.

* Tính tái sử dụng.
* Tính bảo trì.

### Quy tắc nghiệp vụ.

* Khách hàng phải đảm bảo đăng nhập mới có thể gửi đơn đặt hàng
* Khách hàng có thể hủy đơn đặt hàng khi xem tiến trình giao hàng chưa giao tới
* Số lượng tối đa của mỗi sản phẩm là 99

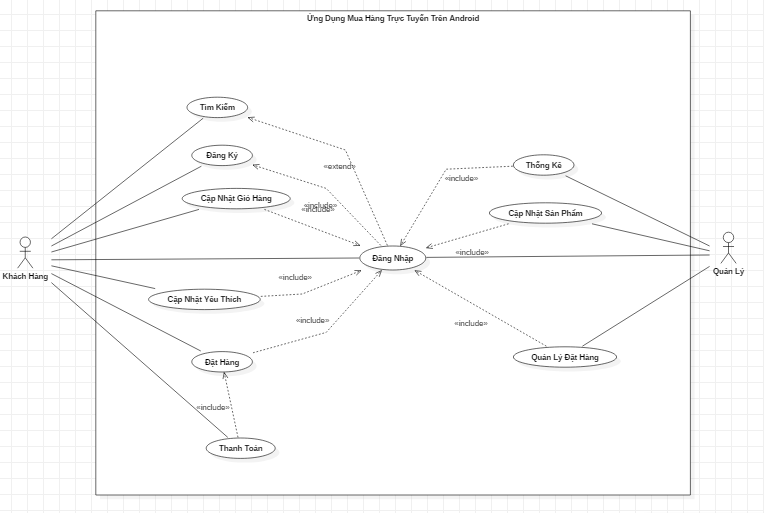
### Quy trình nghiệp vụ.

Quy trình nghiệp vụ của chức năng mua hàng

* Khách hàng cần chọn món hàng cần mua và thêm vào giỏ hàng
* Sau khi đã có giỏ hàng thì khách hàng phải đăng nhập để tiếp tục
* Sau khi đăng nhập cần nhập đầy đủ thông tin giao hàng để xác nhận
* Cuối cùng là theo dõi tiến trình để nhận hàng.

## . Mô hình hóa yêu cầu chức năng.

### Sơ đồ use case.



### Đặc tả use case.

#### 1.3.2.1. Đặc tả UC đăng ký

* Tác nhân: khách hàng
* Mô tả: cho phép khách hàng đăng kí làm thành viên của hệ thống
* Tiền điều kiện:
* Luồng sự kiện chính:

1. Khách xem chọn đăng ký
2. Form đăng kí thành viên hiển thị
3. Khách hàng nhập thông tin cá nhân cần thiết vào form đăng ký
4. Nhấn nút đăng ký
5. Hệ thống cập nhật thông tin của khách hàng vào danh sách thành viên. Nếu thông tin nhập không chính xác thì thực hiện luồng nhánh A1. Nếu nhập chính xác thì thực hiện bước 6
6. Hệ thống cập nhật thông tin của khách hàng vào danh sách thành viên
7. UC Kết thúc

* Luồng sự kiện rẽ nhánh :

Luồng nhánh A1: Quá trình nhập thông tin không chính xác

1. Hệ thống thông báo thông tin nhập không chính xác
2. Hệ thống yêu cầu khách hàng nhập lại thông tin

3. Nếu khách hàng đồng ý thì quay lại bước 2 của luồng sự kiện chính, nếu không đồng ý thì UC kết thúc.

* Hậu điều kiện: Khách hàng trờ thành thành viên của hệ thống

#### 1.3.2.2. Đặc tả UC đăng nhập.

* Tác nhân: Khách hàng
* Mô tả: UC cho phép đăng nhập vào hệ thống
* Tiền điều kiện: khách hàng chưa đăng nhập vào hệ thống
* Luồng sự kiện chính:

1. Khách hàng chọn chức năng đăng nhập
2. Form đăng nhập hiện ra
3. Nhập tên tài khoản, mật khẩu vào form đăng nhập
4. Hệ thống kiểm tra tên, mật khẩu của thành viên
5. Nếu việc đăng nhập thành công thì .... Nếu khách hàng nhập sai tên, mật khẩu thì chuyển sang luồng nhánh A1
6. UC kết thúc

Luồng rẽ sự kiện chính:

Luồng nhánh A1: Khách hàng đăng nhập không thành công

1. Hệ thống thông báo quá trình đăng nhập không thành công

2. Hệ thống yêu cầu nhập lại tên đăng nhập và mật khẩu

3. Nếu đồng ý thì quay về bước 2 của luồng sự kiện chính , nếu không đồng ý thì UC kết thúc hoặc nếu khách hàng quên mật khẩu thì có thể chọn chức năng quên mật khẩu và nhập đầy đủ thông tin hệ thống yêu cầu để hệ thống đổi lại mật khẩu.

* Hậu điều kiện: Khách hàng đã đăng nhập vào hệ thống thành công và có thể sử dụng một số chức năng ràng buộc đăng nhập mà hệ thống cung cấp

#### 1.3.2.3. Đặc tả UC tìm kiếm.

* Tác nhân: Khách Hàng
* Mô tả: Cho phép khách hàng tìm kiếm món hàng cần tìm
* Tiền điều kiện: Sản phẩm tìm kiếm phải hiển thị trên hệ thống
* Luồng sự kiện chính:

1. Khách hàng chọn chức năng tìm kiếm
2. Khách hàng xem thông tin chi tiết món hàng cần tìm
3. UC kết thúc

* Hậu điều kiện: Khách hàng tìm được món hàng mình cần tìm

#### 1.3.2.4. Đặc tả UC cập nhật sản phẩm trong giỏ hàng.

* Tác nhân: Khách Hàng
* Mô tả: Cho phép khách hàng thêm sản phẩm cần mua vào giỏ hàng và xóa ra khỏi giỏ hàng
* Tiền điều kiện: khách hàng phải đăng nhập
* Luồng sự kiện chính:

1. Thêm giỏ hàng:
   1. Hệ thống hiển thị thông tin chi tiết của sản phẩm cần đưa vào giỏ hàng
   2. Nhấn chức năng thêm vào giỏ hàng để đưa sản phẩm vào lưu trữ trong giỏ hàng.
2. Xóa sản phẩm ra khỏi giỏ hàng
   1. Khách hàng chọn sản phẩm cần xóa khỏi giỏ hàng
   2. Chỉnh sửa số lượng sản phẩm về 0
   3. Hệ thống hiển thị thông báo khách hàng xác nhận. Ngược lại thực hiện luồng sự kiện rẽ nhánh A1
   4. Nếu chấp nhận thì sản phẩm sẽ bị xóa.
   5. Hệ thống hiển thị đã loại bỏ thành công
3. UC kết thúc

* Luồng sự kiện rẽ nhánh:

Luồng nhánh A1: khách hàng hủy xóa sản phẩm và quay lại giỏ hàng

* Hậu điều kiện: thông tin sản phẩm đã được đưa vào giỏ hàng phải được lưu trữ hoặc sản phẩm bị xóa ra khỏi giỏ hàng

#### 1.3.2.5. Đặc tả UC cập nhật sản phẩm trong yêu tích.

* Tác nhân: Khách Hàng
* Mô tả: Cho phép khách hàng thêm sản phẩm vào yêu thích
* Tiền điều kiện: khách hàng phải đăng nhập
* Luồng sự kiện chính:

A. Thêm sản phẩm vào yêu thích

1.1. Hệ thống hiển thị thông tin những sản phẩm cần đưa vào yêu thích

1.2. Nhấn chức năng thêm vào yêu thích để đưa sản phẩm đã chọn vào lưu trữ trong yêu thích.

B. Xóa sản phẩm ra khỏi yêu thích

2.1. Khách hàng chọn sản phẩm cần xóa khỏi yêu thích

2.2. Nhấn chức năng xóa yêu thích để thực hiện loại bỏ

2.3. Nếu sản phẩm đã được chọn, hệ thống hiển thị thông báo khách hàng đã xóa khỏi yêu thích.

2.4. Hệ thống đã loại bỏ thành công khỏi danh sách yêu thích

3. UC kết thúc

* Luồng sự kiện rẽ nhánh:
* Hậu điều kiện: thông tin sản phẩm đã được đưa vào yêu thích phải được lưu trữ hoặc sản phẩm bị xóa ra khỏi yêu thích

#### 1.3.2.6. Đặc tả UC đặt hàng.

* Tác nhân: Khách hàng
* Mô tả: Gửi đơn đặt hàng những món hàng đã lưu trong giỏ hàng đến quản lý
* Tiền điều kiện: Đã đăng nhập vào hệ thống và có sản phầm trong giỏ hàng
* Luồng sự kiện chính:

1. Khách hàng chọn chức năng đặt hàng trong giao diện giỏ hàng
2. Hệ thống hiện thị tổng tiền của giỏ hàng
3. Nếu đồng ý mua hàng khách hàng sẽ tiếp nhận thông tin cần nhập: họ tên, số điện thoại, địa chỉ giao hàng , thời gian giao hàng để hoàn thành
4. Chọn đặt hàng. Nếu thông tin cần nhập đúng thì thực hiện bước 5. Nếu sai thì thực hiện luồng sự kiện rẽ nhánh A1
5. Hệ thống thông báo đặt hàng thành công.

* Luồng sự kiện A1: đơn hàng nhập sai

1. Hệ thống thông báo tạo đơn đặt hàng không thàng công
2. Hệ thống hiển thị lại form nhập thông tin và các trường hợp nhập sai
3. Khách hàng nhập lại thông tin, nếu muốn ngưng giao dịch thì thực hiện luồng sự kiện rẽ nhánh A2
4. Quay lại bước 4 của luồng sự kiện chính

* Luồng sự kiện A2: Khách hàng hủy đơn đặt hàng

1. Khách hàng nhấn hủy bỏ đơn hàng
2. Quay lại bước 2 luồng chính
3. UC kết thúc

* Hậu điều kiện: đơn hàng được gửi cho admin

#### 1.3.2.7. Đặc tả UC quản lý dơn đặt hàng.

* Tác nhân: Quản lý
* Mô tả: UC cho phép người quản lý nhận các thông tin về đơn đặt hàng của khách hàng
* Tiền điều kiện: đơn hàng đã được tạo ra
* Luồng sự kiện chính:

1. Hệ thống hiển thị có đơn hàng mới
2. Người quản lý chọn xem đơn hàng mới
3. Hệ thống hiển thị thông tin về những đơn hàng
4. UC kết thúc

* Hậu điều kiện: kiểm soát được những đơn đặt hàng

#### 1.3.2.8. Đặc tả UC cập nhật sản phẩm.

* Tác nhân: Quản lý
* Mô tả: UC cho phép người quản lý thêm, xóa, thay đổi thông tin của các sản phẩm
* Tiền điều kiện: người quản lý phải đăng nhập vào hệ thống
* Luồng sự kiện chính:

1. Người quản lý chọn kiểu tác động lên sản phẩm: thêm sản phẩm , thay đổi thông tin sản phẩm, xóa sản phẩm ra khỏi danh mục

A. Thêm sản phẩm:

1.1 Hệ thống hiển thị form nhập thông tin sản phẩm người quản lý nhập thông tin sản phẩm.

1.2 Nhấn nút lưu thông tin.

1.3 Nếu nhập thành công thì thực hiện bước 1.5(lưu thông tin sản phẩm). Nếu sai thực hiện luồng sự kiện rẽ nhánh A1.

1.4. Nếu sản phẩm nhập đã tồn tại thực hiện luồng sự kiện A2. Nếu đúng thực hiện bước tiếp theo

1.5. Lưu thông tin sản phẩm.

B. Thay đổi thông tin sản phẩm:

2.1 Hệ thống hiển thị form sửa thông tin của sản phẩm

2.2 Người quản lý nhập các thông tin cần thay đổi

2.3 Nhấn nút lưu thông tin

2.4 Nếu việc thay đổi thành công thì thực hiện bước tiếp theo

2.5. Nếu sai thực hiện luồng sự kiện rẽ nhánh A1

2.6 Lưu thông tin sản phẩm

C. Xóa sản phẩm

3.1 Người quản lý chọn sản phẩm cần xóa

3.2 Nhấn nút xóa để thực hiện loại bỏ sản phẩm

3.3 Hệ thống hiển thị thông báo xác nhận loại bỏ. Nếu người quản lý đồng ý thì thực hiện bước tiếp theo. Nếu không đồng ý thì UC kết thúc

3.4 Thông báo sản phẩm đã được xóa

3.5 Hệ thống hiển thị lại danh sách sản phẩm

2. UC kết thúc

- Luồng sự kiện rẽ nhánh A1:

1. Hệ thống thông báo việc nhập dữ liệu không hợp lệ

2. Người quản lý nhập lại thông tin

3. Quay lại Bước 1.2

- Hậu điều kiện: các thông tin về sản phẩm được cập nhật vào cơ sở dữ liệu.

- Luồng sự kiện rẽ nhánh A2:

1. Dữ liệu sản phẩm nhập vào đã có trong hệ thống hoặc không trùng 1 số dữ liệu

2. Hệ thống tự động cập nhật dữ liệu mới

3. Quay lại bước 1.5

#### 1.3.2.9. Đặc tả UC thống kê.

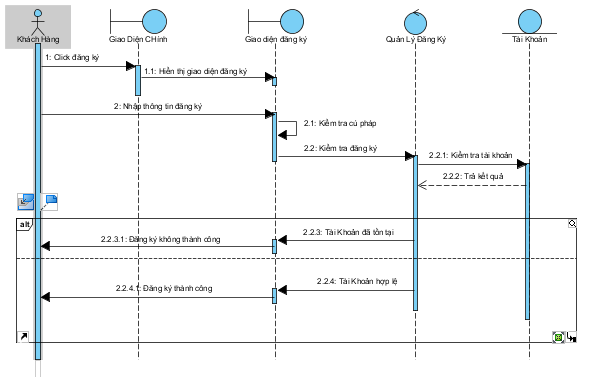
* Tác nhân: Quản lý
* Mô tả: UC cho phép người quản lý xem thống kê doanh số
* Tiền điều kiện: đã có đơn hàng thành công
* Luồng sự kiện chính:

1. Người quản lý chọn xem thống kê
2. Hệ thống hiển thị thông tin thống kê doanh số
3. UC kết thúc

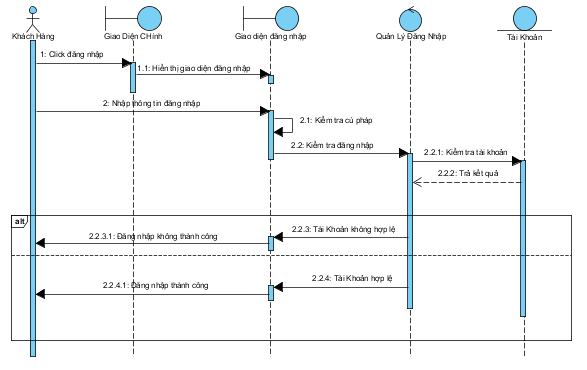
* Hậu điều kiện: kiểm soát được doanh thu

### Sơ đồ sequence.

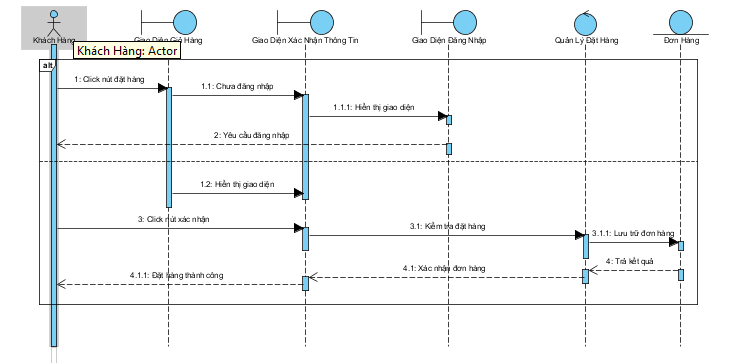
#### 1.3.3.1. Sơ đồ sequence đăng ký.



#### 1.3.3.2. Sơ đồ sequence đăng nhập.

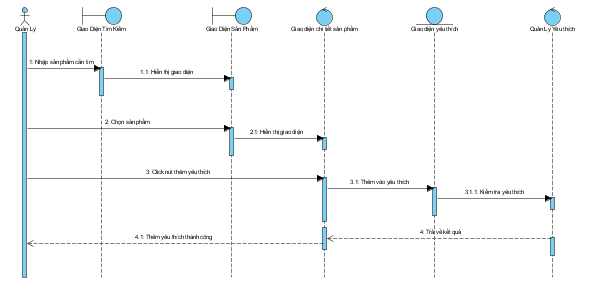


#### 1.3.3.3. Sơ đồ sequence đặt hàng.

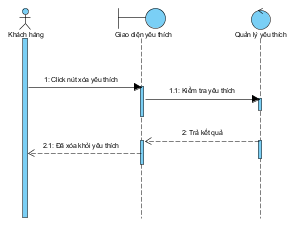


#### 1.3.3.4. Sơ đồ sequence cập nhật yêu thích.

#### 1.3.3.4.1. Sơ đồ sequence thêm yêu thích.

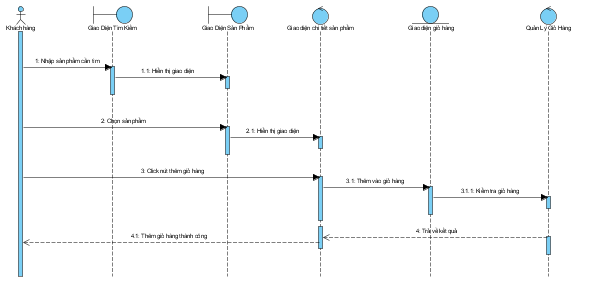


#### 1.3.3.4.2. Sơ đồ sequence xóa yêu thích.

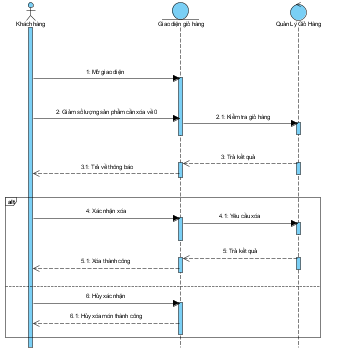


#### 1.3.3.5. Sơ đồ sequence cập nhật giỏ hàng.

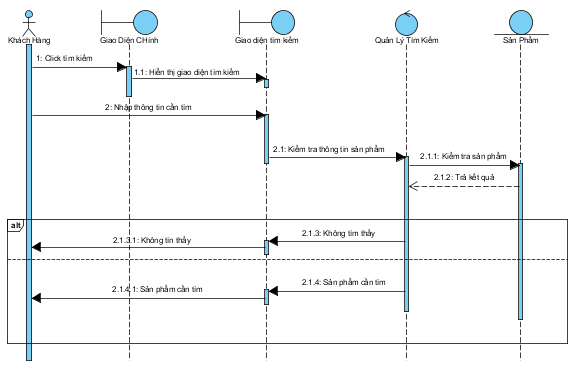
#### 1.3.3.5.1. Sơ đồ sequence thêm giỏ hàng.



#### 1.3.3.5.2. Sơ đồ sequence xóa giỏ hàng.

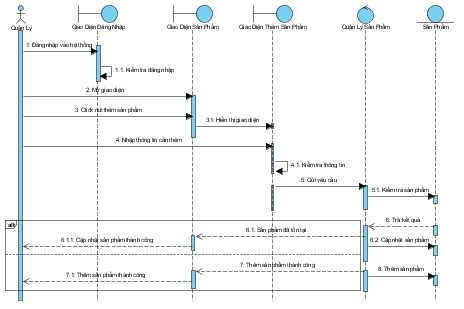


#### 1.3.3.6. Sơ đồ sequence tìm kiếm.

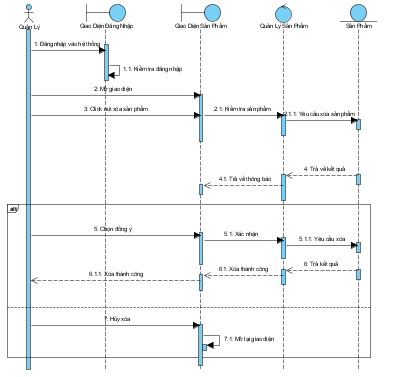


#### 1.3.3.7. Sơ đồ sequence cập nhật sản phẩm.

#### 1.3.3.7.1. Sơ đồ sequence thêm sản phẩm.

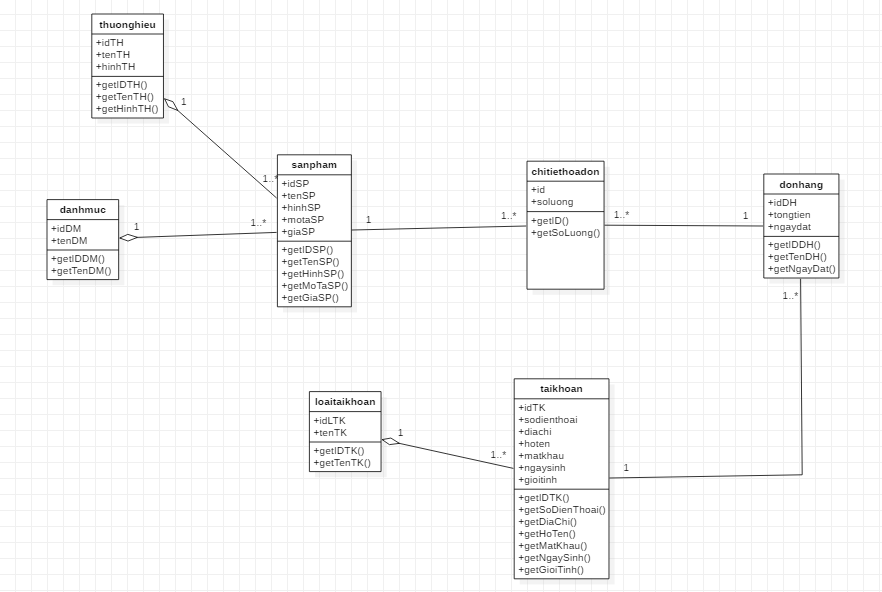


#### 1.3.3.7.2. Sơ đồ sequence xóa sản phẩm.



#### 1.3.3.8. Sơ đồ sequence thống kê.

### Sơ đồ class.



# CHƯƠNG 2. Cơ Sở Dữ Liệu

## 2.1. Giới thiệu tổng quan về Firebase.

### Firebase là gì?

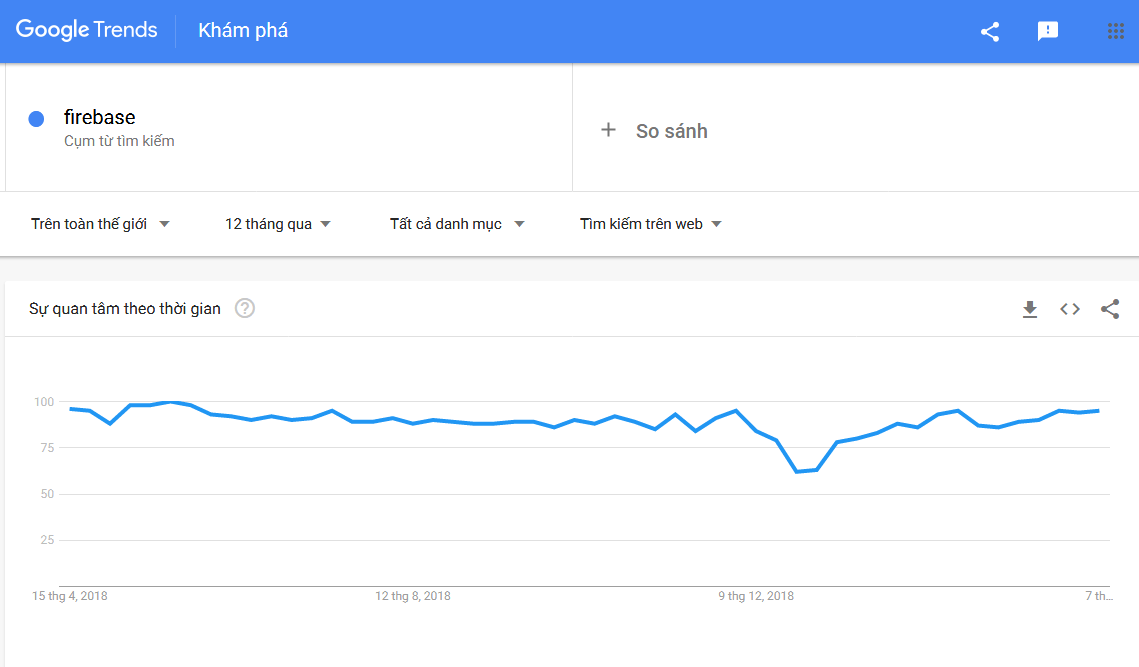
Firebase là một nền tảng phát triển ứng dụng di động và web. Họ cung cấp rất nhiều công cụ và dịch vụ để phát triển ứng dụng chất lượng, rút ngắn thời gian phát triển và phát triển cơ sở người dùng mà không cần quan tâm đến hạ tầng phần cứng.

Firebase là sự kết hợp giữa nền tảng cloud với hệ thống máy chủ cực kì mạnh mẽ của Google. Firebase cung cấp cho chúng ta những API đơn giản, mạnh mẽ và đa nền tảng trong việc quản lý, sử dụng database.

Ví dụ: Với Firebase, chúng ta có thể tự xây dựng một ứng chat thời gian thực như Facebook mesage, Zalo… trong vài ngày, thậm chí vài giờ.

Đơn giản vì giờ chúng ta chỉ cần phải lo phát triển phía Client( Ứng dụng mobile), còn phần backend(server) đã có Firebase lo, chúng ta chỉ cần gọi API là đủ.

Trên thế giới thì xu hướng sử dụng Firebase rất lớn. Không tin, các bạn check Google Trend sẽ rõ.



### 2.1.2. Lịch sử phát triển của FireBase.

Firebase có tiền thân là Envolve. Đây là dịch vụ cung cấp những API để bạn dễ dàng tích hợp tính năng chat vào trang web.

Điều thú vị là người dùng Envolve sử dụng dịch vụ để truyền dữ liệu chứ không đơn thuần cho ứng dụng chat. Họ sử dụng Envolve để đồng bộ dữ liệu của những ứng dụng như game online, danh bạ, lịch…

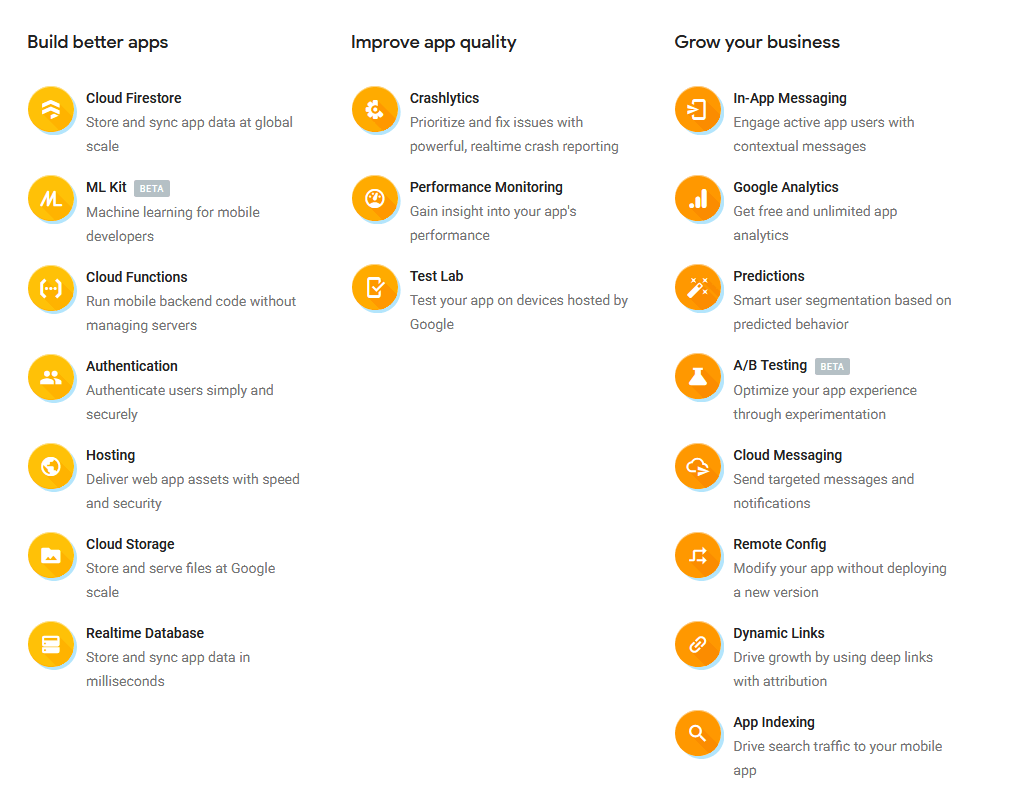
Nhận biết được điều này, CEO của Envolve đã tách biệt hệ thống chat và đồng bộ dữ liệu thời gian thực thành 2 mảng riêng biệt.

Đến năm 2012, Firebase được thành lập như một công ty cung cấp dịch vụ **Backend-as-a-Service** theo thời gian thực.

Ngửi thấy mùi tiềm năng, năm 2014, Google lập tức mua lại Firebase với giá không được tiết lộ. Và giờ Google phát triển Firebase thành một dịch vụ đồ sộ như bạn đang thấy.

### Những dịch vụ nổi bật

Hiện nay, danh mục dịch vụ của Firebase rất nhiều. Từ hệ thống chat thời gian thực, đến A/B testing… và cả ML KIT( Bộ công cụ phát triển Machine learning).



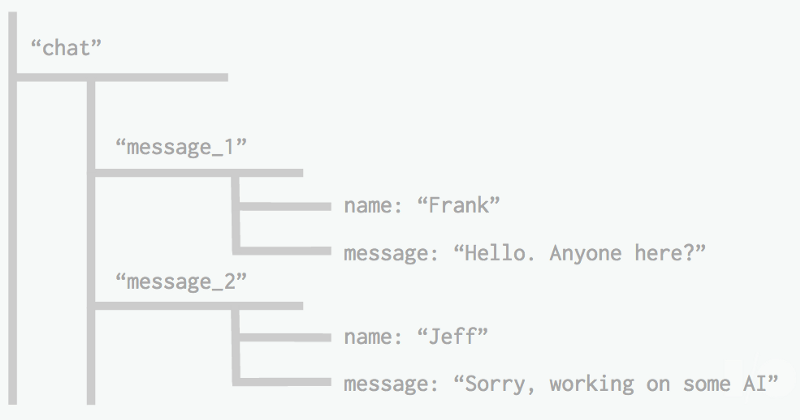
*Các dịch vụ của Firebase*

Chúng ta cùng điểm qua một số dịch vụ nổi bật, hay được sử dụng nhất nhé!

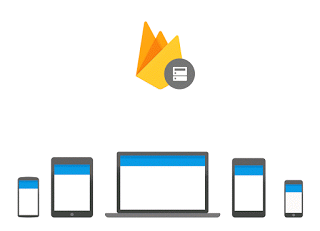
### Realtime Database

Firebase Realtime Database là kiểu dữ liệu NoSQL được lưu trữ trên cloud, cho phép bạn lưu trữ và đồng bộ dữ liệu người dùng theo thời gian thực.

Thực chất, dữ liệu của bạn được lưu dưới dạng JSON object, và developer có thể quản lý theo thời gian thực.



Như hình minh họa  trên, chỉ với một API duy nhất, bạn sẽ có được cả dữ liệu mới nhất và cả những bản update của nó.

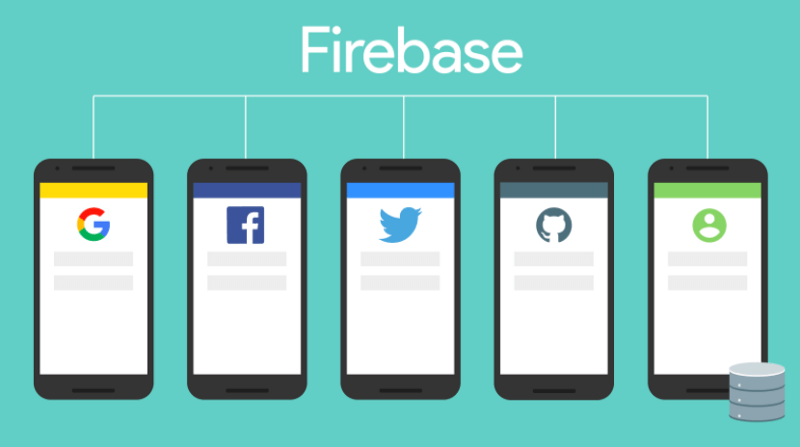


Realtime syncing( đồng bộ theo thời gian thực) giúp người dùng truy cập vào dữ liệu của họ ở bất kỳ thiết bị nào.

Có một ưu điểm của Realtime Database là Firebase sẽ cung cấp cho bạn một bộ SDK để bạn dễ dàng xây dựng ứng dụng mobile, web mà không cần một server.

Khi thiết bị ngoại tuyến(offline), Realtime Database SDK sẽ sử dụng bộ nhớ của thiết bị. Ứng dụng vẫn tương tác với người dùng như bình thường. Đến khi thiết bị online trở lại, nó tự động đồng bộ lên server.

### Authenticate



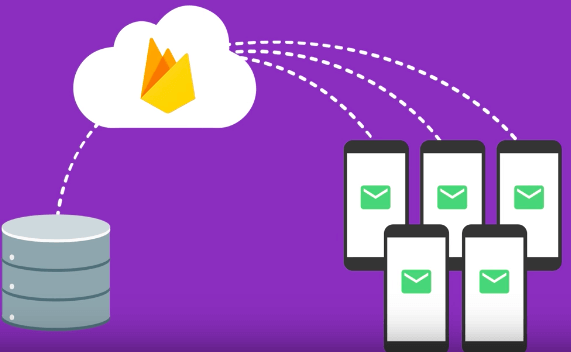
Với tính năng này của Firebase, bạn sẽ dễ dàng xây dựng tính năng login mà không cần phải sử dụng dữ liệu đăng ký riêng. Trước mình cũng có bài viết hướng dẫn các bạn sử dụng tính năng này để [xây dựng tính năng xác thực bằng React native](https://vntalking.com/react-native-authentication-voi-firebase-don-gian-va-de-hieu.html).

Firebase cung cấp một số phương pháp authenticate cho ứng dụng của bạn như:

* Email & Password
* Phone numbers
* Google
* Facebook
* Twitter
* Và còn nhiều hơn nữa!

Có một số ứng dụng nổi tiếng sử dụng tính năng này của Firebase như: Google Sign-in, Smart Lock, và [Chrome Password Manager](https://passwords.google.com/).

### Firebase Cloud Messaging(FCM).



Đây chính là tính năng khởi thủy của Firebase, giúp chúng ta xây dựng ứng dụng chat -trò chuyện. Giờ đây, nó còn cho phép bạn đẩy thông báo( push notification) tới nhiều thiết bị Android, IOS hay Web.

Bạn có thể gửi thông báo(tối đa 2KB) hay tin nhắn( giới hạn 4KB) với độ tin cậy cao và được tối ưu cho Battery.

Với FCM, bạn có thể tích hợp với Firebase Analytics để bạn có thể dễ dàng phân tích người dùng, từ đó đưa ra chiến lược marketing hợp lý.

### Firebase Database Query

Bình thường, khi các bạn làm việc với database như MySQL, SQL Server… bạn muốn lấy dữ liệu ra thì sẽ cần phải query vào database với câu lệnh SQL rất phực tạp. Fireabase database query giúp đơn giản hóa quá trình này.

Firebase có 4 kiểu sắp xếp:

* orderByKey()
* orderByChild(‘child’)
* orderByValue()
* orderByPriority()

Hay như một số câu lệnh query lấy dữ liệu có giới hạn như:

* startAt(‘value’)
* endAt(‘value’)
* equalTo(‘child\_key’)
* limitToFirst(10)
* limitToLast(10)

Mình sẽ thử so sánh với SQL truyền thống nhé.

Với SQL, để thực hiện một câu lệnh query: Bạn sẽ cần chọn column từ một table nào đó. Ví dụ: mình chọn cột Users.

Sau đó bạn thực hiện truy vấn với những điều kiện mong muốn qua câu lệnh WHERE. Như ví dụ, mình muốn lấy tất cả Users có tên là “Anh Son”.

* **SELECT** \* **FROM** Users **WHERE** **name** === "Anh Son";

Nếu muốn giới hạn số lượng Users được lấy ra thì thêm câu lệnh LIMIT.

* **SELECT** \* **FROM** Users **WHERE** **name** === "GeekyAnts" LIMIT 10;

Trong Firebase, việc query cũng hoàn toàn tương tự:

* const db = firebase.database();
* const firebaseRef = db.child('child');
* firebaseRef.orderByChild("users").equalTo("GeekyAnts ") .on("child\_added",
* function(snapshot){
* console.log(snapshot.key)
* });

### Remote Config



Thông thường, các ứng dụng sẽ có mục Setting được lưu trữ ở từng thiết bị. Nếu người dùng mà gỡ ứng dụng thì lần sau cài đặt lại, ứng dụng sẽ mất các thiết lập trước đó.

Hoặc trường hợp khác, nếu người dùng cài đặt ứng dụng trên nhiều thiết bị khác nhau. Họ sẽ phải tự cấu hình thiết lập bằng tay trên từng thiết bị sao cho giống nhau.

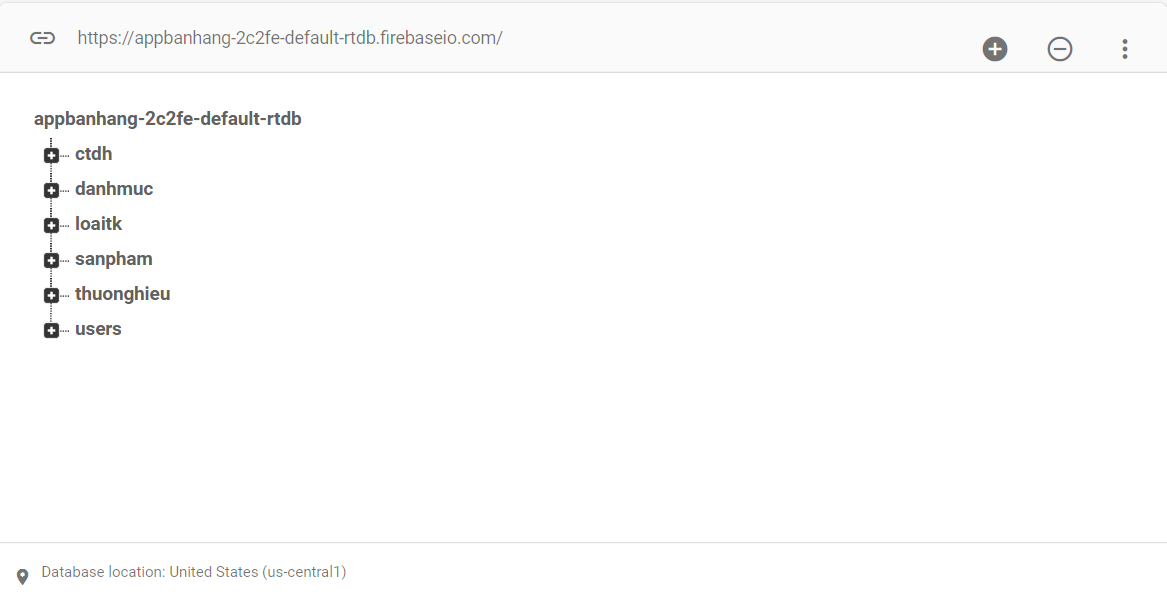
Về phía nhà phát hành, nếu bạn muốn cập nhập thiết lập ứng dụng từ xa như: đến một ngày nào đặc biệt nào đó( Nghỉ lễ tết, ngày quốc tế phụ nữ…), bạn muốn đổi màu ứng dụng, muốn hiển thị Promotion nào đó để quảng cáo…

Không lẽ bạn phải bắt người dùng phải cập nhật ứng dụng để xem quảng cáo?

Và [Remote Config](https://vntalking.com/firebase-remote-config-cau-hinh-app-tu-xa-va-con-hon-the-nua.html) là giải pháp giúp bạn thực hiện điều đó. Bạn sẽ chỉ phải cập nhật trên Fireabase mà ứng dụng phía người dùng sẽ tự đổi theo.

## Cơ sở dữ liệu của ứng dụng.

### Tổng quan cơ sở dữ liệu của ứng dụng.

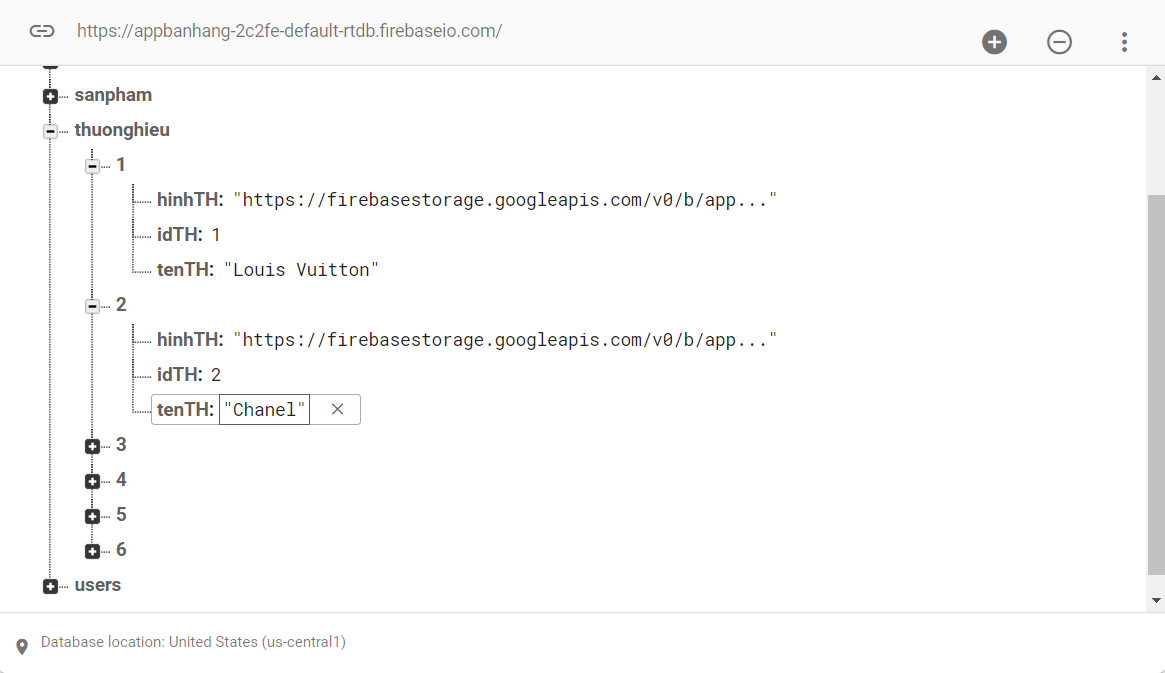


### Một số thành phần dữ liệu

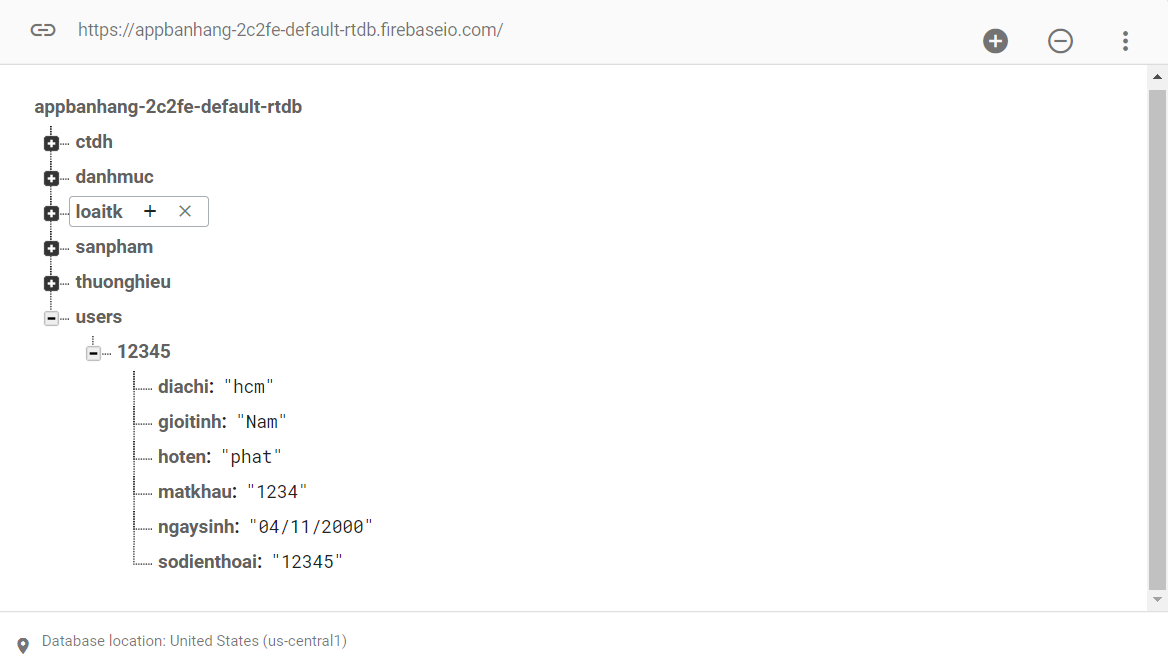
### Dữ liệu danh mục.



### Dữ liệu thương hiệu.



### Dữ liệu tài khoản.



### Dữ liệu sản phẩm.



## Mã hóa dữ liệu

### Tại sao không nên lưu plain-text, encrypt hoặc dùng MD5, SHA-1

Nếu lưu plain-text, database bị hack, SQL-injection, password user chìa ra theo một cách không thể dễ dàng hơn để đánh cắp.

Nếu mã hóa 2 chiều, sẽ luôn có một cách để giải mã bằng một chìa-khóa nào đó, sẽ phải tìm cách lưu chìa-khóa một cách an toàn.

MD5 và SHA-1 được chứng minh có đụng độ, nghĩa là 2 password khác nhau, khi hash bằng MD5 hoặc SHA-1 có thể ra cùng một chuỗi.

### Tại sao phải Salt

Ta đã biết, hash algorithm là one-way-function, tức là không thể suy ngược trực tiếp ra password nếu có hash\_value (khác với mã hóa, có thể giải mã thông qua chìa-khóa).

Tuy nhiên vẫn có cách để từ hash\_value có thể suy gián tiếp ra được password ví dụ brute-force attach, dictionary attach -> điểm chung là ta cần **thử**và **đoán** password nhiều lần cho tới khi đúng cái cần tìm.

Một cách khác đó là ta có thể **tính toán trước** giá trị hash của tất cả các trường hợp và của tất cả các thuật toán -> cách này khó, tốn thời gian, nhưng bây giờ với tốc độ tính toán của máy tính, ta vẫn có thể làm được. Bảng lưu trữ password + hash\_value của password gọi là **Raibow table**, có thể tự tạo hoặc tải một số bản miễn phí hoặc trả tiền để mua. Từ bây giờ nếu ta có hash\_value ta có thể mapping để suy ra được password.

Tuy nhiên nếu ta chỉ hash mỗi password, ta gặp vấn đề đó là:

* 2 password giống nhau (user vô tình trùng password) thì chuỗi hash(password) sẽ giống nhau.
* User cố tình đặt password đơn giản và phổ biến (ví dụ password < 4 kí tự, toàn số, toàn chữ) -> dễ nhớ cho user nhưng dễ tra ngược.

Và nếu chỉ hash password thì nếu mất hash\_value, có thể tra trong rainbow table để tìm ra được password của người dùng.

Giờ ta thử thay vì hash(password) ta sẽ hash(salt + password):

Từ md5(123456)

id | hash\_md5 |

*---------------------------------------*

1 | e10adc3949ba59abbe56e057f20f883e |

Thành md5(7nWZLcCK0vsPzIM + 123456)

id | hash\_md5 | salt |

*---------------------------------------------------------*

1 | 0510210d4b370165658bdc0d0b005244 | 7nWZLcCK0vsPzIM |

Giờ giả sử, ta mất bảng dữ liệu gồm hash\_md5, salt, kẻ tấn công sẽ phải tính toán lại rainbow table của tất cả các trường hợp cộng với salt. Nếu salt là random cho từng user, kẻ tấn công sẽ phải tính toán toàn bộ trường hợp cộng với riêng từng salt cho toàn bộ user.

Chi phí cho 2 phép tính trên là vô cùng lớn và tốn rất nhiều thời gian để thực hiện. Vậy tóm lại mục đích của salt và random-salt là:

* Bảo vệ user kể cả khi user dùng password phổ biến và password không mạnh vì user không thể nhớ được các password phức tạp nhưng tốc độ tính toán của máy tính thì càng ngày càng nhanh.
* Tạo ra nhiều chi phí tính toán, kẻ tấn công không thể tính toán trước rainbow table.

=> Ta trả lời đc 3 câu hỏi:

* Salt có thể lưu trong database, cùng với hash\_value.
* Không cần tìm cách giữ bí mật salt, nhưng cũng đừng tự ý công khai salt.
* Nhưng bắt buộc phải random salt cho từng user.

### Hash ở đâu.

Giờ giả sử hash(password) ở client-side thì vấn đề là gì?

* Biết được thuật toán dùng để hash.
* Salt sẽ phải sinh ra ở client, vì ta cần hash password với salt (hash(salt + password)), và db chỉ lưu kết quả hash, không lưu salt.
* Nhưng nếu salt sinh ra ở client và salt random thì làm sao để compare với hash\_value trong database? Vì lần chứng thực tới, salt sẽ lại random và sẽ khác với kết quả trong database -> salt phải duy nhất cho tất cả các trường hợp.
* Hoặc salt có thể lưu ở DB, nhưng server phải gửi salt về trước cho user trước khi thực hiện hash -> dễ dàng biết được salt hơn.

Nhìn sơ thì thấy việc dùng duy nhất một salt đã chống lại luận điểm ở mục số 2. Vậy quy trình chứng thực đúng là như thế nào?

* User sẽ gửi plain-text password lên server và over HTTPs.
* Server sẽ kiểm tra trong database lấy ra salt của user đó, cộng chuỗi ta được salt + password.
* Thực hiện hash(salt + password) trên server side.
* Compare kết quả trên với hash\_md5 trong database.

### Tại sao dùng bcrypt thay cho SHA-512

Kết quả của SHA-512 có độ dài 128 kí tự, độ dài của key là 64 bytes. Trông có vẻ cũng khá chắc chắn, vậy tại sao OWASP recommend sử dụng PBKDF2, bcrypt hoặc scrypt hơn là SHA2?

SHA2 là hash algorithm (tất nhiên), nó được thiết kế với mục tiêu là tốc độ, với các CPU hiện đại, có thể generate hàng triệu kết quả trên giây. Nếu dùng một thuật toán có tốc độ như SHA2 tức là bạn đã đem lợi điểm tới cho kẻ tấn công brute-force. Thuật toán nhanh + cấu hình server mạnh, việc brute-force càng trở lên nhanh chóng hơn.

Trong khi đó, bcrypt được gọi là slow-hash algorithm, bcrypt() mất 100ms để tính toán ra chuỗi hash, chậm hơn 10.000 lần so với sha1().

Có nghĩa là vẫn đạt được mục đích hash nhưng giảm thiểu nguy cơ tấn công brute-force.

**Tóm lại**: SHA-512 không phải là một thuật toán yếu, mà vấn đề là SHA-512 không phù hợp cho việc hash password. Nếu cần hash password thì ta nên dùng các thuật toán slow hash như PBKDF2, bcrypt và scrypt.

### Tại sao cần pepper

Một thực tế là nếu bạn chỉ có "muối" mà không có "tiêu", ăn thịt gà luộc sẽ không ngon :v. Giả sử, database bạn chạy RAID-1, một ổ cứng hư và cần thay một ổ cứng mới. Nhưng như ta biết, đĩa bị hư là mirror của đĩa còn lại, bạn phải tiêu hủy ổ cứng hư đó nếu không ai đó có thể lục thùng rác và tái tạo lại một phần dữ liệu trong đĩa hư đó.

Xin lưu ý, bạn cần wipe trước khi vứt bỏ một ổ cứng có dữ liệu dù cá nhân hay server, tuy nhiên đĩa bị sốc điện, bad-sector thì wipe cũng chưa đủ an toàn, tốt nhất nên ngiền ra bã.

Dù random-salt đã làm tăng chi phí tạo ra rainbow table nhưng đời không biết đâu mà lần, kẻ tấn công luôn có những động lực không tưởng để đạt được cái mình muốn. Giả sử kẻ tấn công có một siêu siêu máy tính và một mirror ổ cứng hư lục từ một cái thùng rác nào đó. Với siêu máy tính đó, ta có rainbow-table để tra ngược ra password cần tìm.

Vậy làm sao để giảm thiểu nguy cơ trên? Nguyên tắc là không bỏ tất cả trứng trong một giỏ, đó là pepper. Pepper là một chuỗi tương tự như salt, nhưng khác biệt là ta cần giữ bí mật pepper, lưu ở một chỗ khác ngoài database, và không cần pepper-per-user, chỉ cần 1 pepper là đủ.

Từ

hash(salt + password)

Thành

hash(pepper + salt + password)

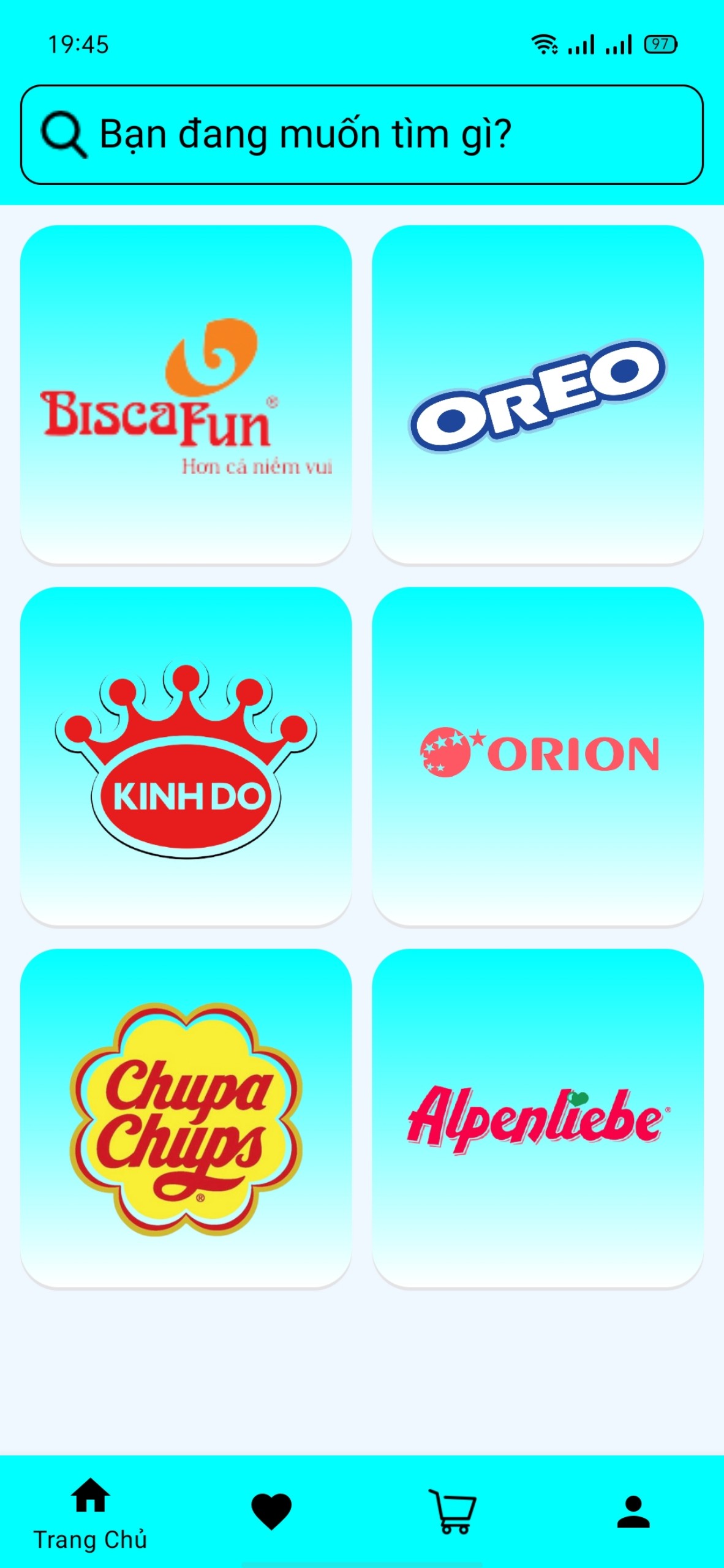
Ta nên lưu pepper ở application hoặc ở một service khác, nếu database bị compromise, thì kẻ tấn công cũng không có pepper để tạo ra rainbow-table.

\

# CHƯƠNG 3. Kết Quả Thực Nghiệm.

## 3.1. Thiết kế giao diện hệ thống.

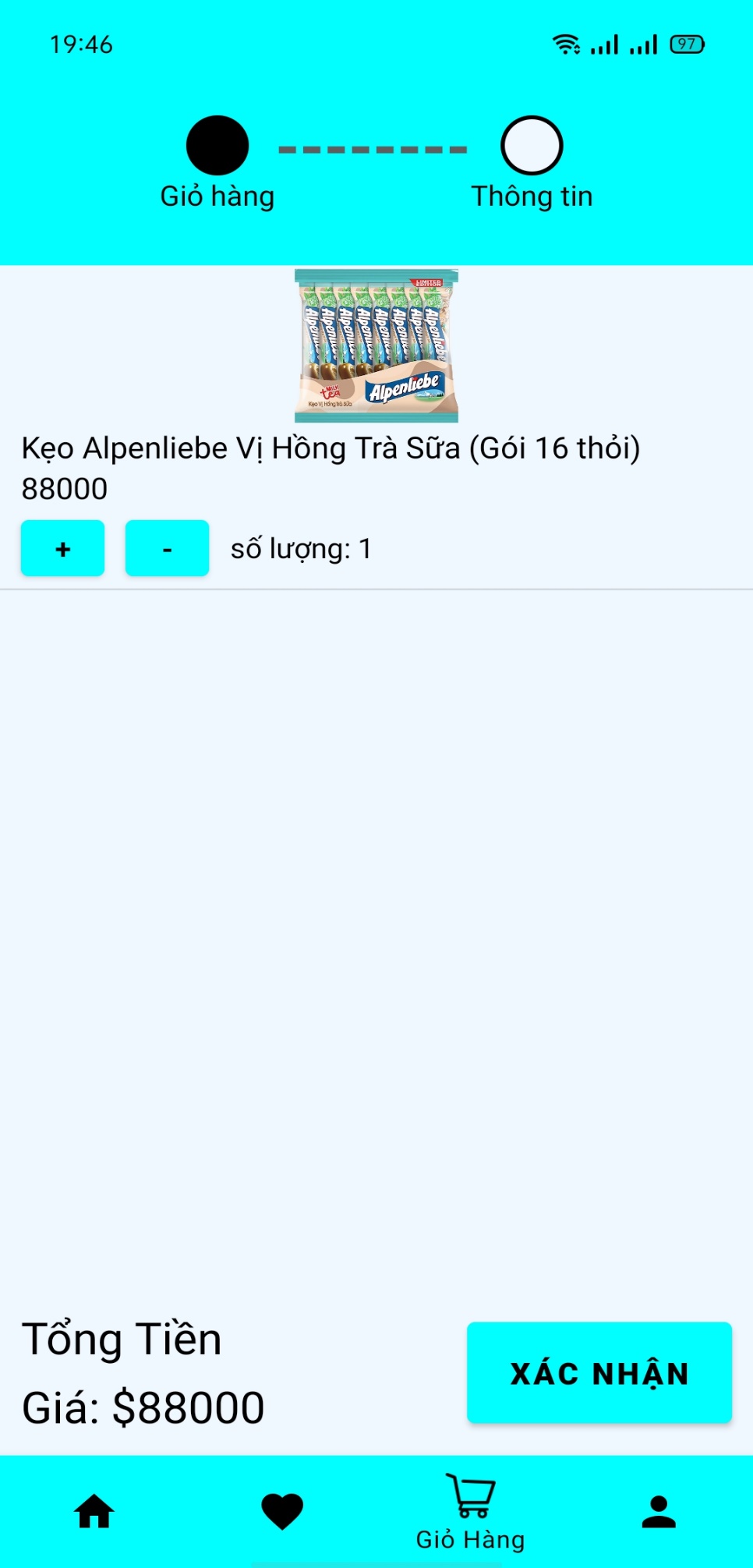
### Giao diện trang chủ



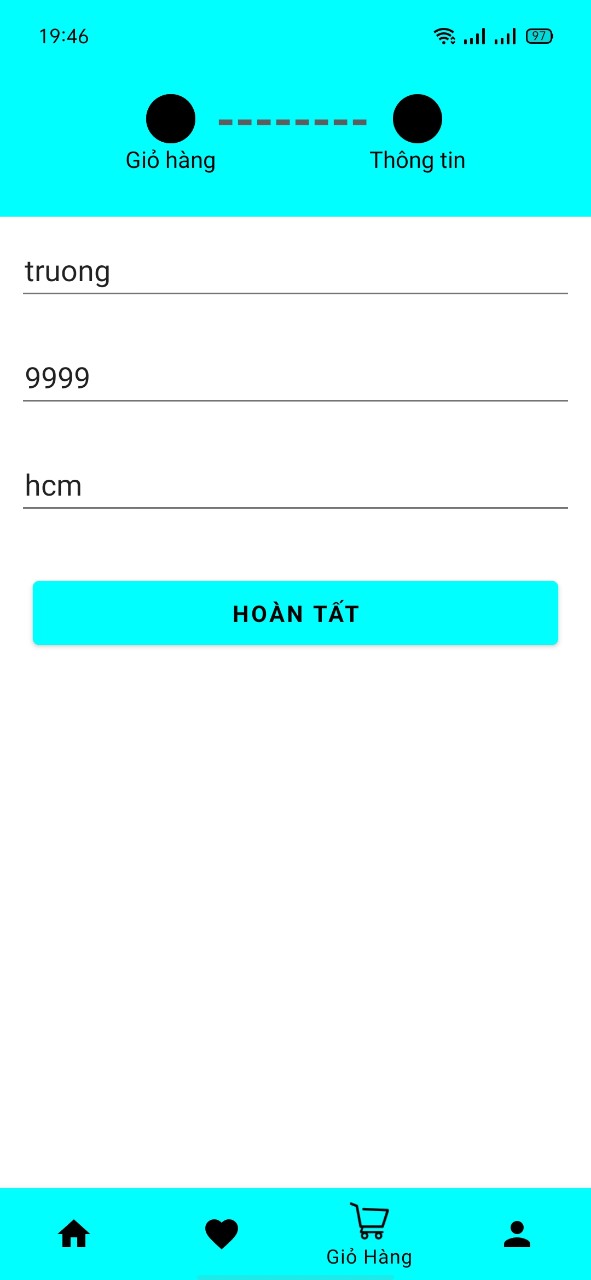
### Giao diện yêu thích



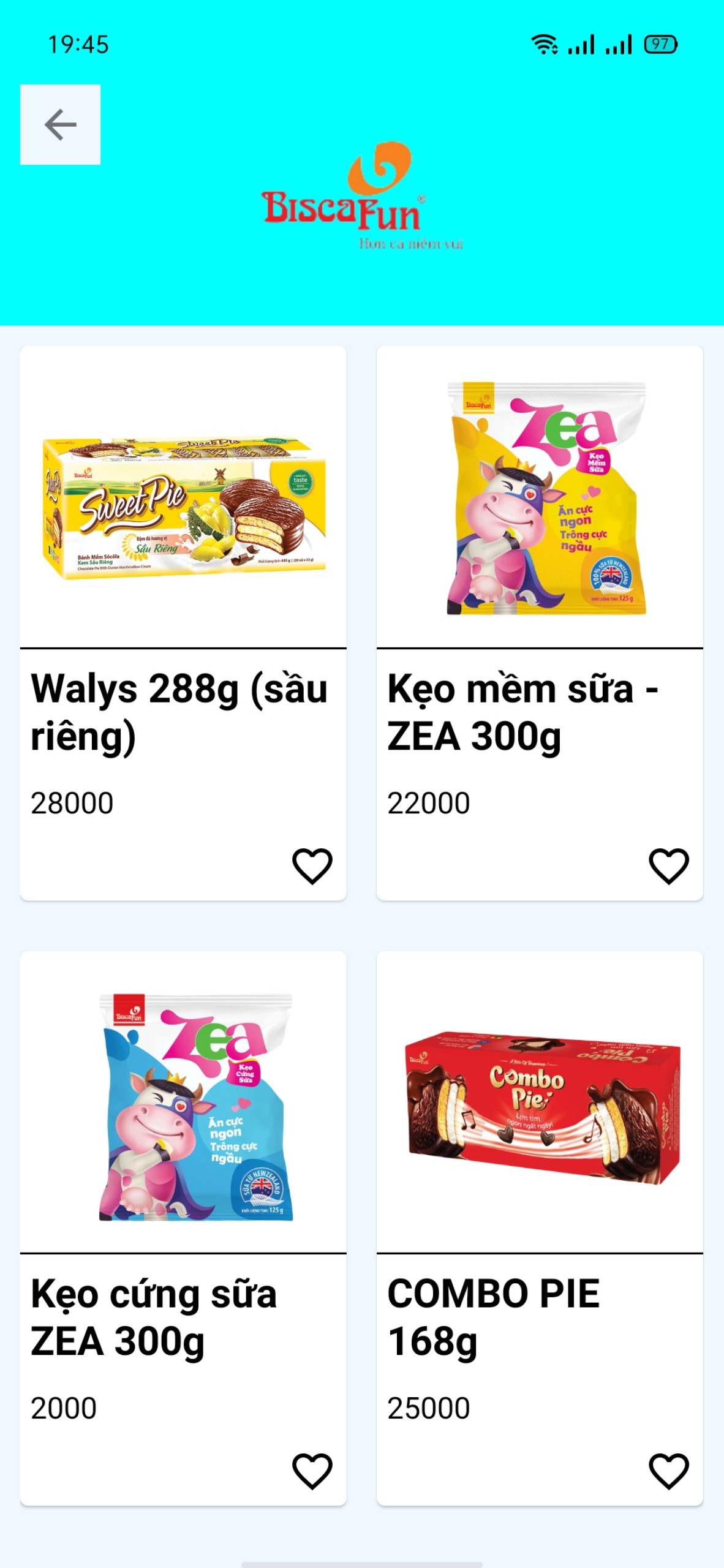
### Giao diện giỏ hàng



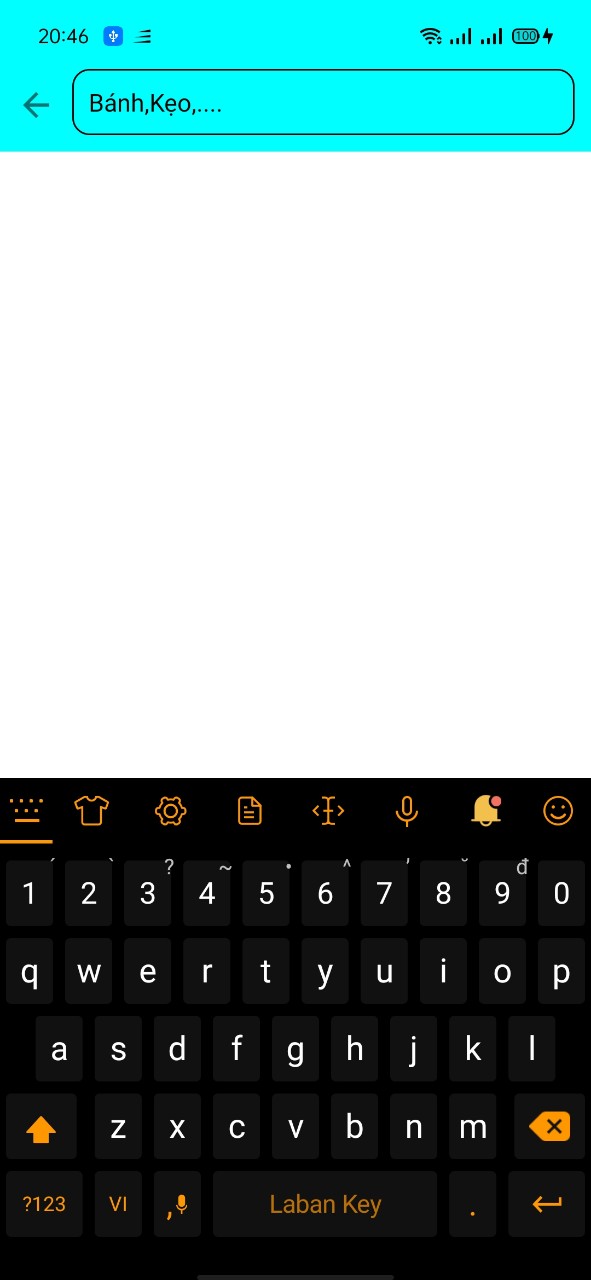
### Giao diện xác nhận đơn hàng



### Giao diện sản phẩm



### Giao diện tìm kiếm



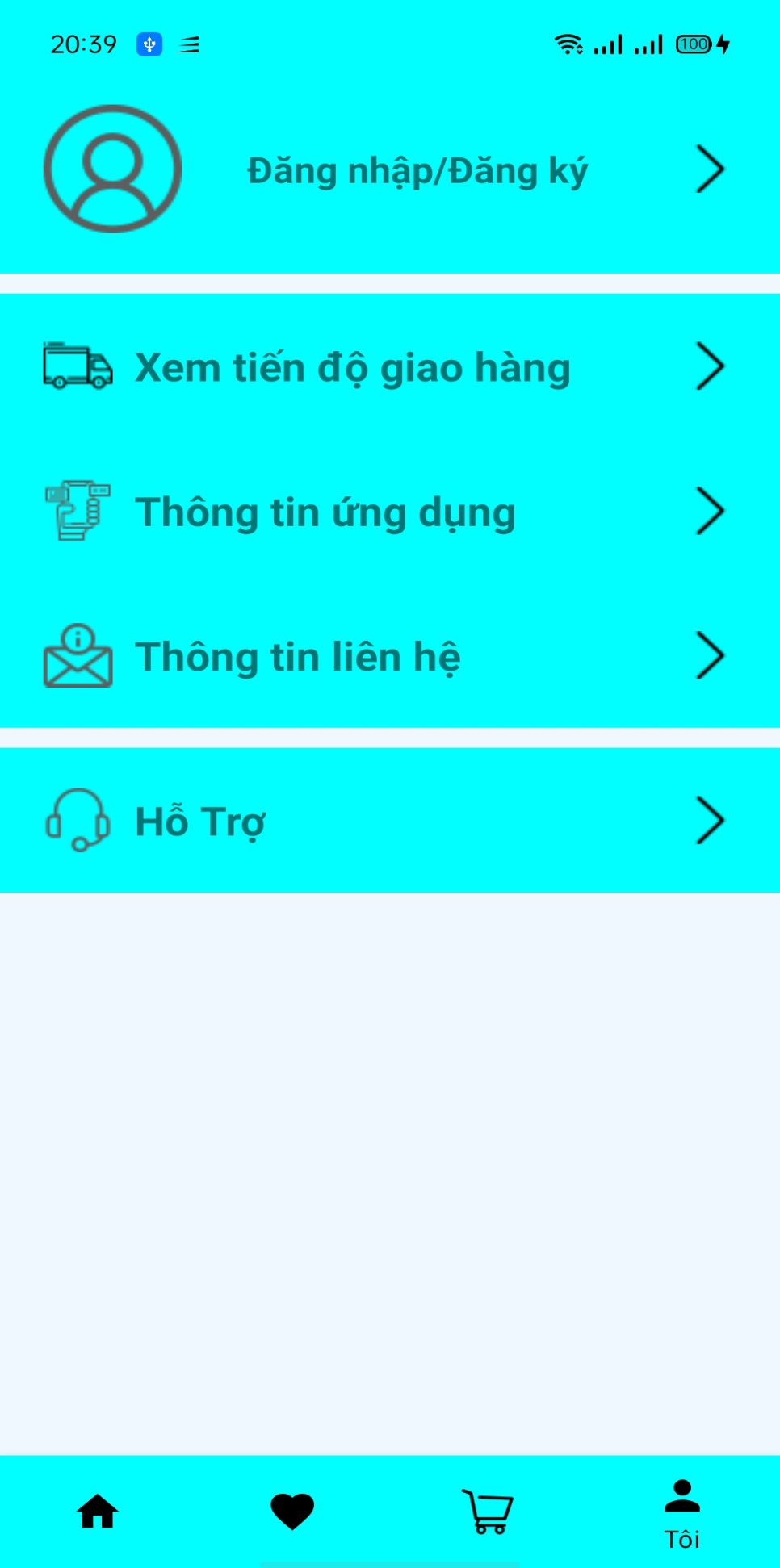
### Giao diện đăng nhập



### Giao diện đăng ký



### Giao diện tài khoản cá nhân



## 3.2. Testcase của ứng dụng.

|  |  |  |
| --- | --- | --- |
| Chức năng | Input | Output |
|  | Nhập họ tên vượt quá 14 – 28 kí tự, chứa kí tự đặc biệt | Họ tên không hợp lệ |
|  | Nhập số điện thoại lớn hơn 10 kí tự và không bắt đầu bằng số 0 | Số điện thoại không hợp lệ |
| Đăng Ký | Nhập mật khẩu vượt quá 14 – 28 kí tự | Mật khẩu không hợp lệ |
|  | Nhập địa chỉ vượt quá 100 kí tự | Địa chỉ không hợp lệ |
|  | Nhập không đầy đủ thông tin hoặc thiếu một trường dữ liệu bất kì | Vui lòng nhập đầy đủ thông tin |
|  | Để trống tên đăng nhập, mật khẩu hoặc cả 2 | Vui lòng nhập đầy đủ thông tin |
| Đăng Nhập | Nhập sai tên tài khoản | Tên tài khoản không hợp lệ |
|  | Nhập sai mật khẩu | Sai mật khẩu |
| Tìm kiếm | Nhập icon, kí tự đặc biệt, số | Hiển thị không tìm thấy kết quả |
|  | Khách hàng chưa đăng nhập | Hiển thị giao diện yêu cầu đăng nhập |
| Đặt Hàng | Trong giỏ hàng chưa có sản phẩm | Thông báo giỏ hàng chưa có sản phẩm |
|  | Số lượng sản phẩm không vượt quá 99 | Hệ thống chặn thêm số lượng |
| Yêu Thích | Nhấn vào yêu thích của sản phẩm đã dược yêu thích | Xóa khòi yêu thích |
|  |  |  |